Fast Recursive Filters for Simulating Nonlinear Dynamic Systems

van Hateren, Johannes

Published in:
Neural computation

Publication date:
2008

LETTER Communicated by Markus Diesmann

Fast Recursive Filters for Simulating Nonlinear Dynamic Systems

J. H. van Hateren
j.h.van.hateren@rug.nl
Netherlands Institute for Neuroscience, Royal Netherlands Academy of Arts and Sciences, Amsterdam, and Institute for Mathematics and Computing Science, University of Groningen, The Netherlands

A fast and accurate computational scheme for simulating nonlinear dynamic systems is presented. The scheme assumes that the system can be represented by a combination of components of only two different types: first-order low-pass filters and static nonlinearities. The parameters of these filters and nonlinearities may depend on system variables, and the topology of the system may be complex, including feedback. Several examples taken from neuroscience are given: phototransduction, photopigment bleaching, and spike generation according to the Hodgkin-Huxley equations. The scheme uses two slightly different forms of autoregressive filters, with an implicit delay of zero for feedforward control and an implicit delay of half a sample distance for feedback control. On a fairly complex model of the macaque retinal horizontal cell, it computes, for a given level of accuracy, one to two orders of magnitude faster than the fourth-order Runge-Kutta. The computational scheme has minimal memory requirements and is also suited for computation on a stream processor, such as a graphical processing unit.

1 Introduction

Nonlinear systems are ubiquitous in neuroscience, and simulations of concrete neural systems often involve large numbers of neurons or neural components. In particular, if model performance needs to be compared with and fitted to measured neural responses, computing times can become quite restrictive. For such applications, efficient computational schemes are necessary. In this letter, I present such a highly efficient scheme, which has recently been used for simulating image processing by the primate outer retina (van Hateren, 2006, 2007). The scheme is particularly suited for data-driven applications, where the time step of integration is dictated by the sampling interval of the analog-to-digital or digital-to-analog conversion. It assumes that the system can be decomposed into components of only two types: static nonlinearities and first-order low-pass filters. Interestingly, these components are also the most common ones used in neuromorphic

VLSI circuits (Mead, 1989). In the scheme presented here, the components need not have fixed parameters but are allowed to depend on the system state. They are arranged in a possibly complex topography, typically involving several feedback loops. The efficiency of the scheme is produced by using very fast recursive filters for the first-order low-pass filters. I will show that it is best to use slightly different forms of the filter algorithm for feedforward and feedback processing loops.

No attempt is made to rigorously analyze the convergence or optimality of the scheme, which would be difficult to do for arbitrary nonlinear systems. The scheme should therefore be viewed as a practical solution that works well for the examples I give here but may need specific testing and benchmarking on new problems.

The scheme I present can be efficiently implemented on stream processors. Recently there has been growing interest in using such processors for high-performance computing (e.g., Göddeke, Robert Strzodka, & Turek, 2007; Ahrenberg, Benzie, Magnor, & Watson, 2006; Guerrero-Rivera, Morrison, Diesmann, & Pearce, 2006). In particular, the arrival of affordable graphical processing units (GPUs) with raw computating power more than an order of magnitude higher than that of CPUs is driving this interest (see http://www.gpgpu.org). Current GPUs typically have about 100 processors that can work in parallel on data in the card's memory. Once the data and the (C-like) programs are loaded into the card, the card computes essentially independent of the CPU. Results subsequently can be uploaded to the CPU for further processing. GPUs are especially suited for simulating problems, such as in retinal image processing, that can be written as parallel, local operations on a two-dimensional grid.

Stream processors are, unlike CPUs, data driven and not instruction driven. They process the incoming data as they become available and therefore usually need algorithms with fixed, or at least predictable, computing times. The processing scheme I present in this letter has a fixed computing time. Moreover, it has low computational cost and low memory requirements, because it deals only with current and previous values of input, state variables, and output. The output is produced without delays that are not part of the model, that is, at the same time step as the current input, and the scheme is thus also suited for real-time applications.

The letter is organized as follows. First, I present a fairly complete overview of methods to simulate a first-order low-pass filter with a minimal recursive filter. Subsequently, I give several examples of how specific neural systems—in particular, several subsystems of retinal processing and spike generation following the Hodgkin-Huxley equations—can be decomposed into suitable components. Computed results of the various forms of recursive filters are compared with benchmark calculations using a standard Matlab solver. I show that for a practical, fairly complex model, the most efficient algorithm (modified Tustin) outperforms a conventional fourth-order Runge-Kutta integration by one to two orders of

magnitude. Finally, I discuss the merits and limits of the approach taken here.

2 Discrete Simulation of a First-Order Low-Pass Filter

Much of the material presented in this section is not new. However, I found that most of it is scattered throughout the literature, and I therefore give a fairly complete overview. Table 1 summarizes the filters and their properties.

In the continuous time domain, the equation

$$\frac{dy}{dt} + \frac{1}{\tau}y = \frac{1}{\tau}x \tag{2.1}$$

describes a first-order low-pass filter transforming an input function $x(t)$ into an output function $y(t)$, where τ is the time constant and the coefficient in front of x is chosen such that the filter has unit DC gain: $y = x$ if the input is a constant. In the examples below, I usually write this equation in the standard form:

$$\tau\dot{y} = x - y. \tag{2.2}$$

Fourier-transforming this equation gives as the transfer function of this filter

$$H(\omega) = \frac{\tilde{y}}{\tilde{x}} = \frac{1}{1 + i\omega\tau}, \tag{2.3}$$

where the tilde denotes Fourier transforms. The impulse response of the filter is

$$\begin{aligned} h(t) &= \tfrac{1}{\tau}e^{-t/\tau} \quad \text{for } t \geq 0 \\ &= 0 \qquad\qquad\ \ \text{for } t < 0. \end{aligned} \tag{2.4}$$

We assume now that $x(t)$ is available only at discrete times $t_n = n\Delta$, as $x_n = x(n\Delta)$, and that we require $y(t)$ at the same times, as $y_n = y(n\Delta)$. Here Δ is the time between samples. In conformation with the most common integration schemes, we further assume that for calculating the current value of the output, only the current value of the input, the previous value of the output, and possibly the previous value of the input are available. We therefore seek real coefficients a_1, b_0, and b_1 such that

$$y_n = -a_1 y_{n-1} + b_0 x_n + b_1 x_{n-1} \tag{2.5}$$

Table 1: Autoregressive Filters Approximating $\tau \dot{y} = x - y$ by $y_n = -a_1 y_{n-1} + b_0 x_n + b_1 x_{n-1}$, with Sample Distance Δ, and $\tau' \equiv \tau/\Delta$.

Scheme	Forward Euler	Backward Euler	Trapezoidal Rule	Exponential Euler	Zero-Order Hold	First-Order Hold	Modified Tustin's Method
Also known as			Tustin's method Bilinear transformation Crank-Nicholson	Exponential integration	Step-invariant approximation Exact integration	Ramp-invariant approximation Triangular rule	
$-a_1$ (weight of y_{n-1}, previous output)	$(1 - 1/\tau')$	$\tau'/(\tau' + 1)$	$(\tau' - 0.5)/(\tau' + 0.5)$	$e^{-1/\tau'}$	$e^{-1/\tau'}$	$e^{-1/\tau'}$	$(\tau' - 0.5)/(\tau' + 0.5)$
b_0 (weight of x_n, present input)	–	$1/(\tau' + 1)$	$0.5/(\tau' + 0.5)$	–	$1 - e^{-1/\tau'}$	$1 - \tau' + \tau' e^{-1/\tau'}$	$1/(\tau' + 0.5)$
b_1 (weight of x_{n-1}, previous input)	$1/\tau'$	–	$0.5/(\tau' + 0.5)$	$1 - e^{-1/\tau'}$	–	$\tau' - (1 + \tau')e^{-1/\tau'}$	–
Implicit delay	$\Delta/2$	$-\Delta/2$	0	$\Delta/2$	$-\Delta/2$	0	$-\Delta/2$
Symbol			τ^0				τ^-
Remarks	Can be unstable		Preferred choice for feedforward				Preferred choice for feedback

produces an output close to that expected from equation 2.2. The indices and signs of the coefficients are chosen here in such a way that they are consistent with common use in the digital processing community for describing IIR (infinite impulse response) or ARMA (autoregressive, moving average) filters that relate the z-transforms of input and output (Oppenheim & Schafer, 1975). I will not use the z-transform formalism here, but only note that Fourier-transforming equation 2.5 and using the shift theorem gives

$$\tilde{y}_n = -a_1 \tilde{y}_n e^{-i\omega\Delta} + b_0 \tilde{x}_n + b_1 \tilde{x}_n e^{-i\omega\Delta}, \tag{2.6}$$

and therefore a transfer function

$$H(\omega) = \frac{\tilde{y}_n}{\tilde{x}_n} = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}}, \tag{2.7}$$

where the operator $z^{-1} = \exp(-i\omega\Delta)$ represents a delay of one sample.

The coefficients a_1, b_0, and b_1 are not independent because of the additional constraint that the filter of equation 2.2 has unit DC gain. A constant input c must then produce a constant output c, thus, equation 2.5 yields $c = -a_1 c + b_0 c + b_1 c$ and therefore

$$-a_1 + b_0 + b_1 = 1. \tag{2.8}$$

Because representing a general continuous system as in equation 2.2 by a discrete system as in equation 2.5 can be only approximate (note that equations 2.3 and 2.7 cannot be made identical), there is no unique choice for the coefficients a_1, b_0, and b_1. Below I give an overview of several possibilities, mostly available in the literature, and discuss their appropriateness for the computational scheme I present. The first three methods discussed below, forward Euler, backward Euler, and the trapezoidal rule, are derived from general methods for approximating derivatives. The other methods discussed are more specialized, dealing specifically with equation 2.2 and differing with respect to how the input signal is assumed to behave between the sampled values.

2.1 Forward Euler. Forward Euler (Press, Teukolsky, Vetterling, & Flannery, 1992) is used quite often in neural simulations. Applied to equation 2.2, it amounts to the approximation

$$y_n \approx y_{n-1} + \dot{y}_{n-1}\,\Delta = y_{n-1} + (x_{n-1} - y_{n-1})\,\Delta/\tau; \tag{2.9}$$

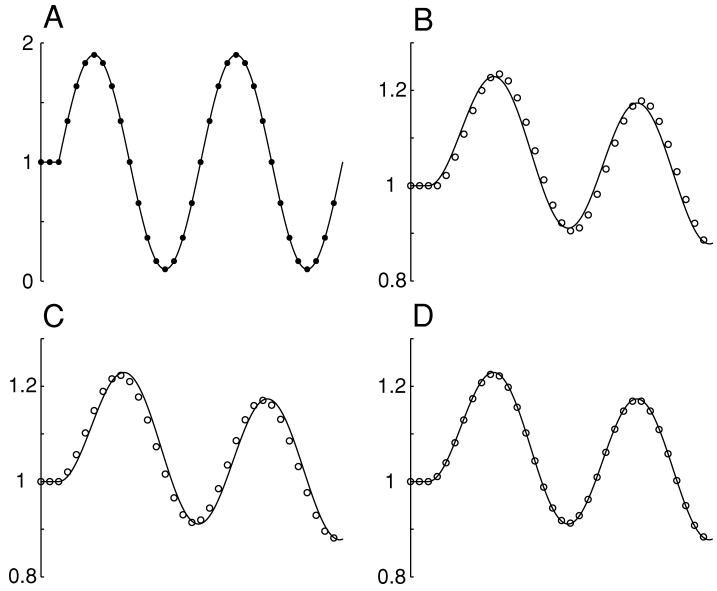

Figure 1: (A) Starting sinusoid (continuous line) and function values at the sample times (filled circles, 16 samples per period). The function equals 1 at times earlier than shown. (B) Continuous line: sinusoid of A filtered by equation 2.2 with $\tau' = 16$, computed with Matlab ode45; open circles: result of filtering the samples of A with equation 2.10, the recurrence equation that follows from forward Euler. Output samples lag by approximately half a sample distance. (C) As B, for backward Euler (see equation 2.12). Output samples lead by approximately half a sample distance. (D) As B, for trapezoidal (see equation 2.14).

hence, we get the recurrence equation,

$$\begin{aligned} y_n &= (1 - 1/\tau')y_{n-1} + (1/\tau')x_{n-1} \\ \text{with} \quad \tau' &= \tau/\Delta. \end{aligned} \tag{2.10}$$

Here as well as below, I use τ', which is τ normalized by the sample distance, to keep the equations concise. Equation 2.10 suffers from two major problems: first, it is not very accurate, and even unstable for small τ' (Press et al., 1992), and second, it produces an implicit delay of $\Delta/2$ for centered samples. The second problem is illustrated in Figure 1. Figure 1A shows a starting sinusoid, where the filled circles give the function values at the sampling times. The continuous function of Figure 1A can subsequently be filtered by equation 2.2 using a standard integration routine (Matlab ode45) at a time resolution much better than Δ (obviously, in this simple case, the

result could have been obtained analytically, but we will encounter other examples below where this is not possible). Figure 1B shows the result (continuous line). When the samples of the sinusoid are processed by equation 2.10, the result lags by half a sampled distance (open circles in Figure 1B).

2.2 Backward Euler. Backward Euler (Press et al., 1992) applied to equation 2.1 yields

$$y_n \approx y_{n-1} + \dot{y}_n \Delta = y_{n-1} + (x_n - y_n)\Delta/\tau, \tag{2.11}$$

hence

$$y_n = [\tau'/(\tau' + 1)]y_{n-1} + [1/(\tau' + 1)]x_n. \tag{2.12}$$

Backward Euler is stable (Press et al., 1992) and slightly more accurate than forward Euler, but suffers from the problem that it produces an implicit delay of $-\Delta/2$ for centered samples, that is, a phase advance. Figure 1C illustrates this, where the continous curve is the correct result (identical curve as the continuous curve in Figure 1B), and the open circles give the result of applying equation 2.12.

2.3 Trapezoidal Rule. The trapezoidal rule (also known as Crank-Nicholson; Rotter & Diesmann, 1999) is equivalent to the bilinear transformation and Tustin's method in digital signal processing (Oppenheim & Schafer, 1975). It combines forward and backward Euler,

$$y_n \approx y_{n-1} + \frac{1}{2}(\dot{y}_{n-1} + \dot{y}_n)\Delta = y_{n-1} + \frac{1}{2}(x_{n-1} - y_{n-1} + x_n - y_n)\Delta/\tau, \tag{2.13}$$

and leads to

$$\begin{aligned} y_n = {} & [(\tau' - 0.5)/(\tau' + 0.5)]y_{n-1} + [0.5/(\tau' + 0.5)]x_n \\ & + [0.5/(\tau' + 0.5)]x_{n-1}. \end{aligned} \tag{2.14}$$

The method is stable and accurate, and it produces a negligible implicit delay (see Figure 1D).

2.4 Exponential Euler. A method that has gained some popularity in the field of computational neuroscience (e.g., in the simulation package Genesis; Bower & Beeman, 1998) is sometimes called exponential integration (MacGregor, 1987; Rotter & Diesmann, 1999) or exponential Euler (Moore & Ramon, 1974; Rush & Larsen, 1978; Butera & McCarthy, 2004). It

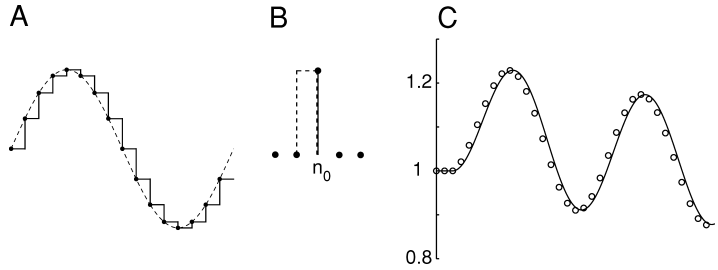

Figure 2: (A) Zero-order hold sampling model, where the sample values (dots) taken from a function (dashed line) are held until a new sample arrives (continuous line). (B) A unit sample (continuous line and filled circles) is assumed here to represent a block in the previous intersample interval (dashed line). (C) Continuous line: sinusoid of Figure 1A filtered by equation 2.2 with $\tau' = 16$, computed with Matlab ode45; open circles: result of filtering the samples of Figure 1A with equation 2.17, the recurrence equation that follows from the ZOH processing scheme (i.e., assumed pulse shape of B).

assumes that the input is approximately constant, namely, equal to x_{n-1}, on the interval from $(n-1)\Delta$ to $n\Delta$. Equation 2.1 then has the exact solution (see, e.g., appendix C.6 of Rotter & Diesmann, 1999)

$$y_n = e^{-1/\tau'} y_{n-1} + (1 - e^{-1/\tau'}) x_{n-1}. \tag{2.15}$$

This method is closely related to forward Euler, as a comparison of equations 2.10 and 2.15 shows: for large τ' (time constant large compared with the sample distance), the factors $\exp(-1/\tau') \approx 1 - 1/\tau'$ and $1 - \exp(-1/\tau') \approx 1/\tau'$ approximate those of forward Euler. The exponential Euler method is stable and more accurate than forward Euler for small τ'. However, it has the same implicit delay of $\Delta/2$ as forward Euler (not shown).

2.5 Zero-Order Hold. When analog-to-digital and digital-to-analog converters are used, a choice has to be made for the assumed signal values between the sample times. A simple, practical choice is to keep the value of the last sample until a new sample arrives. This is called a zero-order hold (ZOH), and for a sampled sinusoid, it assumes the continuous line shown in Figure 2A. It involves an implicit delay of $\Delta/2$. Digitally filtering the samples of a ZOH system can compensate for this delay by assuming that a unit sample at $n = n_0$ (continuous line and filled circles in Figure 2B) represents a block as shown by the dashed line in Figure 2B. The coefficients a_1, b_0, and b_1 for approximating equation 2.2 by equation 2.5 can be readily obtained from the response to this pulse; these coefficients then also apply to an arbitrary input signal, because the filter is linear and time invariant.

For samples $n \geq n_0 + 2$, the present and previous input are zero; thus the terms with b_0 and b_1 do not contribute. Because equation 2.4 shows that the output must decline exponentially, we find $-a_1 = e^{-\Delta/\tau} = e^{-1/\tau'}$. For sample $n = n_0$, the previous input and output are zero; thus, the terms with a_1 and b_1 do not contribute. We then find b_0 from the convolution of the block $s(t)$ (dashed line in Figure 2B) with the pulse response $h(t)$ of the filter, evaluated at sample $n = n_0$:

$$b_0 = \int_{-\infty}^{\infty} h(t')\,p(t - t')\,dt' \Big|_{t=n_0\Delta} = \int_0^{\Delta} \frac{1}{\tau} e^{-t'/\tau} \cdot 1\,dt' = 1 - e^{-\Delta/\tau}$$
$$= 1 - e^{-1/\tau'}. \tag{2.16}$$

With equation 2.8, we then find $b_1 = 1 + a_1 - b_0 = 0$. The recurrence equation therefore is

$$y_n = e^{-1/\tau'} y_{n-1} + (1 - e^{-1/\tau'}) x_n. \tag{2.17}$$

Note that the difference with equation 2.15 is that here the current input sample, x_n, is used, whereas in equation 2.15, it is the previous input sample, x_{n-1}. Whereas equation 2.15 implies a delay of $\Delta/2$, the present scheme has a delay of $-\Delta/2$, that is, a phase advance (see Figure 2C).

The filter in equation 2.17 is a special case of a general scheme of representing linear filters by using the matrix exponential (e.g., Rotter & Diesmann, 1999, where it is called exact integration). Such filters are consistent with assuming a ZOH and therefore imply a delay of $-\Delta/2$. Although Rotter and Diesmann do not use a ZOH but rather a function representation using Dirac δ-functions, a delay is implied by the choice of integration interval in their equation 3, which excludes the previous input sample and fully includes the present input sample. Had the integration interval been chosen symmetrical, the δ-functions at the previous and present input samples would each have contributed by one-half, leading to a scheme with $0.5(x_{n-1} + x_n)$ as input, and therefore an implicit delay of 0.

2.6 First-Order Hold. Another choice for the assumed function values between samples is the first-order hold (FOH), where sample values are connected by straight lines. It assumes that a unit sample at $n = n_0$ (the continuous line and filled circles in Figure 3A) represents a triangular pulse as shown by the dashed line in Figure 3A. The method is also called the triangular or ramp-invariant approximation, and is in fact equivalent to assuming that a function can be represented by B-splines of order one (Unser, 1999, 2005). A general derivation of the recurrence relation, also valid for the more general lead-lag system $\tau_y \dot{y} + y = \tau_x \dot{x} + x$ of which equation 2.2 is a

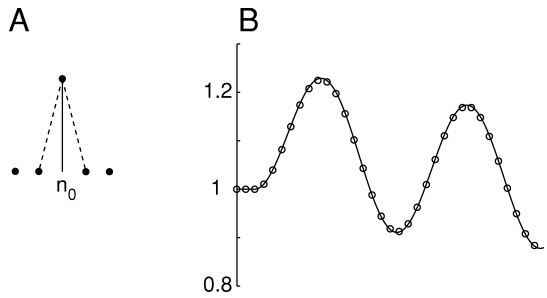

Figure 3: (A) A unit sample (continuous line and filled circle) is assumed here to represent linear interpolation in the previous and next intersample intervals (dashed line). (B) Continuous line: sinusoid of Figure 1A filtered by equation 2.2 with $\tau' = 16$, computed with Matlab ode45; open circles: result of filtering the samples of Figure 1A with equation 2.19, the recurrence equation that follows from the FOH processing scheme (i.e., assumed pulse shape of A).

special case, is given by Brown (2000). A simple alternative derivation goes similarly as given above for the ZOH. For samples $n \geq n_0 + 2$, the present and previous input are zero, and we again find $-a_1 = e^{-1/\tau'}$. For sample $n = n_0$, the previous input and output are zero, and now b_0 equals

$$b_0 = \int_{-\infty}^{\infty} h(t')\,p(t-t')\,dt'\Big|_{t=n_0\Delta} = \int_0^{\Delta} \frac{1}{\tau}\,e^{-t'/\tau}\left(1-\frac{t'}{\Delta}\right)dt'$$
$$= 1 - \tau' + \tau' e^{-1/\tau'}. \tag{2.18}$$

With equation 2.8, we then find $b_1 = 1 + a_1 - b_0 = \tau' - (1+\tau')\exp(-1/\tau')$. The recurrence equation therefore is

$$y_n = e^{-1/\tau'} y_{n-1} + \left(1 - \tau' + \tau' e^{-1/\tau'}\right) x_n + \left(\tau' - (1+\tau')e^{-1/\tau'}\right) x_{n-1}. \tag{2.19}$$

Figure 3B illustrates that the FOH has a negligible implicit delay.

2.7 Centered Step Invariant. The centered step-invariant approximation (e.g., Thong & McNames, 2002) is not often used and is given here only for completeness; its performance is similar to that of FOH and trapezoidal. It assumes that a unit sample at $n = n_0$ represents a block that is, contrary to the regular zero-order hold, centered on the sample time. This is equivalent to assuming that a function can be represented by B-splines of order zero

(Unser, 1999). As before, we must have $-a_1 = e^{-1/\tau'}$, and for b_0 we get

$$b_0 = \int_{-\infty}^{\infty} h(t')p(t-t')\,dt'\Big|_{t=n_0\Delta} = \int_0^{\Delta/2} \frac{1}{\tau} e^{-t'/\tau} \cdot 1\,dt' = 1 - e^{-1/(2\tau')}. \tag{2.20}$$

With equation 2.8, we then find $b_1 = 1 + a_1 - b_0 = \exp(-1/(2\tau')) - \exp(-1/\tau')$. The recurrence equation therefore is

$$y_n = e^{-1/\tau'} y_{n-1} + (1 - e^{-1/(2\tau')})x_n + (e^{-1/(2\tau')} - e^{-1/\tau'})x_{n-1}. \tag{2.21}$$

This method also has a negligible implicit delay (not shown).

2.8 Modified Tustin's Method. Below I show that for implementing nonlinear feedback systems, a delay of $-\Delta/2$ is in fact favorable. One possibility is to use the ZOH for obtaining such a delay, but a modification of Tustin's method (the trapezoidal rule discussed above) is at least as good and has coeffients that are simpler to compute. Whereas the trapezoidal rule has no appreciable implicit delay, because it weighs the present and previous inputs equally ($b_0 = b_1$), it can be given a $-\Delta/2$ delay by combining these weights to apply to the present input only:

$$y_n = [(\tau' - 0.5)/(\tau' + 0.5)]y_{n-1} + [1/(\tau' + 0.5)]x_n. \tag{2.22}$$

The method is evaluated along with the other methods in the remainder of this letter and will be shown to work very well for feedback systems. To my knowledge, this modification of Tustin's method has not been described in the literature before.

3 Relationship Between Recursive Schemes for First-Order Low-pass Filters

A Taylor expansion of the various forms of $-a_1$ gives

$$-a_1 = e^{-1/\tau'} = 1 - \frac{1}{\tau'} + \frac{1}{2\tau'^2} - \frac{1}{6\tau'^3} + \ldots$$

for exponential Euler, ZOH, and FOH, (3.1)

$$-a_1 = 1 - \frac{1}{\tau'} \quad \text{for forward Euler,} \tag{3.2}$$

$$-a_1 = \tau'/(\tau' + 1) = 1/(1 + 1/\tau') = 1 - \frac{1}{\tau'} + \frac{1}{\tau'^2} - \frac{1}{\tau'^3}$$

$+ \ldots$ for backward Euler, (3.3)

$$
\begin{aligned}
-a_1 &= (\tau' - 0.5)/(\tau' + 0.5) = \left(1 - \frac{1}{2\tau'}\right) \Big/ \left(1 + \frac{1}{2\tau'}\right) \\
&= \left(1 - \frac{1}{2\tau'}\right)\left(1 - \frac{1}{2\tau'} + \frac{1}{4\tau'^2} - \frac{1}{8\tau'^3} + \ldots\right) \\
&= 1 - \frac{1}{\tau'} + \frac{1}{2\tau'^2} - \frac{1}{4\tau'^3} \\
&\quad + \ldots \quad \text{for trapezoidal and modified Tustin.}
\end{aligned} \tag{3.4}
$$

Compared to the theoretical exponential decline, equation 2.4, the exponential Euler, ZOH, and FOH are fully correct; the forward and backward Euler schemes are correct only up to the factor with $(1/\tau')$; and trapezoidal and modified Tustin are correct up to the factor with $(1/\tau')^2$. The accuracy of the last is related to the fact that $(\tau' - 0.5)/(\tau' + 0.5)$ is a first-order Padé approximation of $\exp(-1/\tau')$ (Bechhoefer, 2005). Note that in the limit of $\tau \gg \Delta$, all algorithms use approximately the same weight for the previous output sample: $1 - 1/\tau'$.

With respect to the weights acting on the input, the algorithms presented above can be divided into three groups, depending on the implicit delay they carry (see Table 1). If only the previous input sample is used (forward and exponential Euler), there is a delay of $\Delta/2$; if only the present input sample is used (backward Euler, ZOH, and modified Tustin's method), there is a delay of $-\Delta/2$; and if both the previous and present input samples are used (trapezoidal and FOH), there is no delay. Below we analyze only the groups with delays $-\Delta/2$ and 0.

The coefficients b_0 of the group with the phase advance (delay $-\Delta/2$) can be expanded as

$$
b_0 = 1 - e^{-1/\tau'} = \frac{1}{\tau'} - \frac{1}{2\tau'^2} + \frac{1}{6\tau'^3} + \ldots \quad \text{for ZOH} \tag{3.5}
$$

$$
\begin{aligned}
b_0 &= 1/(\tau' + 1) = \frac{1}{\tau'}\frac{1}{(1 + 1/\tau')} \\
&= \frac{1}{\tau'} - \frac{1}{\tau'^2} + \frac{1}{\tau'^3} - \ldots \quad \text{for backward Euler}
\end{aligned} \tag{3.6}
$$

$$
\begin{aligned}
b_0 &= 1/(\tau' + 0.5) = \left(\frac{1}{\tau'}\right) \Big/ \left(1 + \frac{1}{2\tau'}\right) = \left(\frac{1}{\tau'}\right)\left(1 - \frac{1}{2\tau'} + \frac{1}{4\tau'^2} - \ldots\right) \\
&= \frac{1}{\tau'} - \frac{1}{2\tau'^2} + \frac{1}{4\tau'^3} - \ldots \quad \text{for modified Tustin,}
\end{aligned} \tag{3.7}
$$

where we find that ZOH and modified Tustin are more similar to each other than to backward Euler.

Finally, the coefficients of the FOH can be compared with those of trapezoidal:

$$b_0 = 1 - \tau' + \tau' e^{-1/\tau'} = 1 - \tau' + \tau' \left(1 - \frac{1}{\tau'} + \frac{1}{2\tau'^2} - \frac{1}{6\tau'^3} + \dots\right)$$

$$= \frac{1}{2\tau'} - \frac{1}{6\tau'^2} + \dots \quad \text{for FOH} \tag{3.8}$$

$$b_0 = 0.5/(\tau' + 0.5) = \left(\frac{1}{2\tau'}\right) \Big/ \left(1 + \frac{1}{2\tau'}\right) = \left(\frac{1}{2\tau'}\right)\left(1 - \frac{1}{2\tau'} + \dots\right)$$

$$= \frac{1}{2\tau'} - \frac{1}{4\tau'^2} + \dots \quad \text{for trapezoidal} \tag{3.9}$$

and

$$b_1 = \tau' - (1 + \tau') e^{-1/\tau'} = \tau' - (1 + \tau') \left(1 - \frac{1}{\tau'} + \frac{1}{2\tau'^2} - \frac{1}{6\tau'^3} + \dots\right)$$

$$= \frac{1}{2\tau'} - \frac{1}{3\tau'^2} + \dots \quad \text{for FOH} \tag{3.10}$$

$$b_1 = 0.5/(\tau' + 0.5) = \frac{1}{2\tau'} - \frac{1}{4\tau'^2} + \dots \quad \text{for trapezoidal.} \tag{3.11}$$

The coefficients start to differ in the factor with $(1/\tau')^2$. We will see in the examples in section 4 that FOH and trapezoidal perform very similarly on concrete problems.

4 Examples of Nonlinear Dynamic Systems

In this section, I provide several examples of nonlinear dynamic systems that are well suited to be simulated using autoregressive filters of the type discussed above. I show for these examples how the systems can be rearranged to contain only static nonlinearities and first-order low-pass filters. Furthermore, I compare the results of several of the algorithms presented above with an accurate numerical benchmark and discuss the speed and accuracy of the various possibilities.

4.1 Phototransduction: Coupled Nonlinear ODEs. An example of a system where coupled nonlinear differential equations can be represented by a feedback system is the phototransduction system in the cones of the vertebrate retina. I concentrate here on the main mechanism, which provides gain control and control of temporal bandwidth (van Hateren, 2005).

For this purpose, a suitable form is given by

$$\dot{X} = 1/(1 + C^4) - \beta X \tag{4.1}$$

$$\dot{C} = (X - C)/\tau_C . \tag{4.2}$$

The variable β is linearly related to the light intensity and can be considered as the input to the system. The variable X represents the concentration of an internal transmitter of the cone and can be considered as the output of the system because it regulates the current across the cone's membrane. The variable C is an internal feedback variable, proportional to the intracellular Ca^{2+} concentration.

We now rewrite the equations such that they get the form of equation 2.2:

$$\tau_\beta \dot{X} = q/(1 + C^4) - X$$

$$\text{with} \quad \tau_\beta = 1/\beta \quad \text{and} \quad q = 1/\beta \tag{4.3}$$

$$\tau_C \dot{C} = X - C. \tag{4.4}$$

By defining a time constant τ_β (actually not a constant, because it varies with β) and an auxiliary variable q, we see that both equations formally take a form similar to equation 2.2, where q now has the role of input to equation 4.3, with the factor $1/(1 + C^4)$ as a gain. We can thus represent these equations by the system diagram shown in Figure 4A. The boxes containing a τ there represent unit-gain first-order low-pass filters. From the system diagram, it is clear that the divisive feedback uses its own result after that has progressed through two low-pass filters and a static nonlinearity. The following describes the algorithm associated with Figure 4A:

- Assume an initial steady state with $\beta = \beta_0$, and obtain initial values of all variables by solving (analytically or numerically) equations 4.3 and 4.4 for $\dot{X} = 0$ and $\dot{C} = 0$.
- Repeat for each time step:
  Read β as input. Compute a_1, b_0, and b_1 for $\tau_\beta = 1/\beta$, and update X by low-pass filtering it, taking $(1/\beta)/(1 + C^4)$ as input to the filter. Use a precomputed a_1, b_0, and b_1 for τ_C to update C by low-pass filtering it, taking X as input to the filter. Write X as output.

Note that τ_β is obtained from the current value of β. In principle, it might have been based partly on the previous value of β as well, because β changes in the interval between the previous and the current sample. However, for τ_β significantly larger than Δ, this is expected to be a second-order effect, and the changing time constant is therefore treated in the simplest possible way, as described in the algorithm above.

Because at each time step only the result of C that was obtained at the previous time step can be used in the division by $(1 + C^4)$, the feedback path would effectively get an (implicit) extra delay of Δ if calculated following

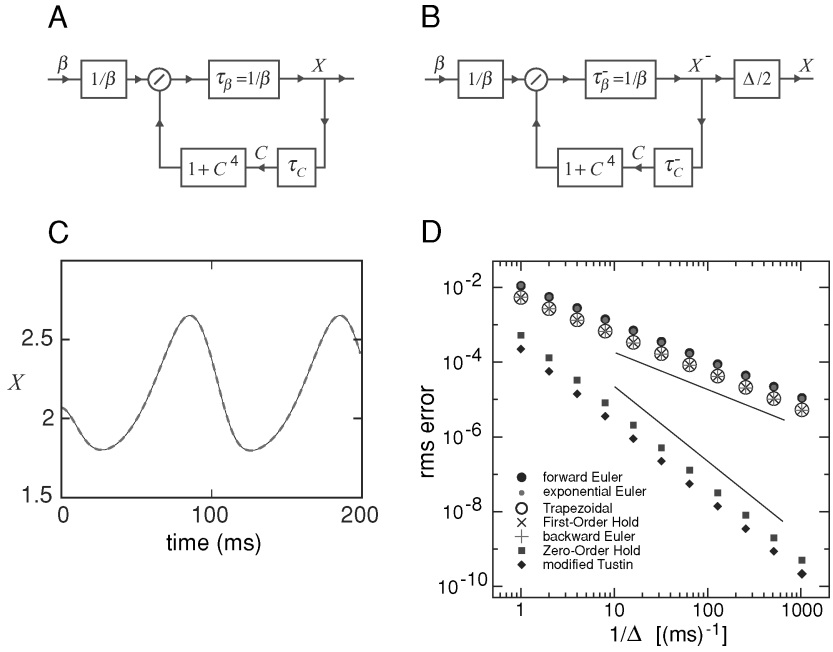

Figure 4: (A) System diagram of equations 4.3 and 4.4. Boxes containing a τ are unit-gain first-order low-pass filters, possibly depending on input or state variables (e.g., τ_β depends on β). The other boxes represent static nonlinearities given by the function definition inside the box. (B) Scheme equivalent to A, where the required phase advance of one sample distance (Δ) for the feedback is obtained by using two low-pass filters of type τ^-, each providing a $-\Delta/2$ delay (i.e., a $\Delta/2$ phase advance). The box to the right represents a $\Delta/2$ delay to compensate for the phase advance of τ_β^-. (C) Thin continuous line: response X of equations 4.3 and 4.4, using $\tau_C = 3$ ms, to $\beta = \beta_0(1 + 0.9\sin(2\pi f t))$ for $t \geq 0$ and $\beta = \beta_0$ for $t < 0$, with $\beta_0 = 0.025\,(\text{ms})^{-1}$ and $f = 10$ Hz, computed with Matlab ode45; dashed line: result of filtering with the scheme of B, with $\Delta = 1$ ms and using the modified Tustin's method for τ^-. (D) Root-mean-square (rms) error between the output when using the various recursive filters for the scheme of B and the result of ode45 at its maximum accuracy setting. Input as in C. The thin, straight lines are an aid for judging the scaling behavior of the various methods, and have slopes of -1 and -2 in double-logarithmic coordinates.

this scheme. Such an extra delay will affect the results (and in extreme cases may lead to spurious oscillations), which can be minimized only by choosing Δ rather small. However, there is a way to alleviate this problem. As we have seen above, several of the autoregressive schemes have an implicit delay of $-\Delta/2$. Because there are two low-pass filters concatenated in the feedback loop, using such a scheme will produce a total delay of

$-\Delta$, exactly compensating for the implicit delay Δ of the feedback. In other words, the divisor used at the point of divisive feedback will have the correct current time. Because the forward low-pass filter, τ_β, has a delay of $-\Delta/2$, we need to compensate that if we require that the output of the system have the right phase. (This may not always be necessary, especially not when the system is part of a larger system, where it would be more convenient to correct the sum of all delays at the final output.) The required delay of $\Delta/2$ can be approximated by linear interpolation, that is, a recurrence equation $y_n = 0.5x_{n-1} + 0.5x_n$. The linear interpolation implies a slight low-pass filtering of the signal and is therefore accurate only if the sampling rate is sufficiently high compared with the bandwidth of the signal. We can then replace the scheme of Figure 4A by the one of Figure 4B, where τ^- indicates that we are using filters with a $-\Delta/2$ delay (see Table 1).

How well do the recursive schemes of section 2 perform on this problem? To evaluate that, the thin, continuous line in Figure 4C shows the response X of equations 4.3 and 4.4 to a sinusoidal modulation of β, computed using the Matlab routine ode45 at high time resolution and high precision settings. The dashed line shows the result when using the scheme of Figure 4B with the modified Tustin's method used for τ^- with $\Delta = 1$ ms. How the accuracy depends on Δ is evaluated in Figure 4D, which shows the rms (root mean square) deviation from the ode45 benchmark as a function of Δ, not only for the modified Tustin's method but also for most of the other schemes. To get a fair comparison, Figure 4A was used for schemes with implicit delays 0 and $\Delta/2$, where for the latter, an explicit delay of $-\Delta/2$ was added as a final stage. As is clear, the ZOH and especially the modified Tustin's method are superior. They scale more favorably as a function of $1/\Delta$, and for a given level of accuracy it is sufficient to use a Δ at least an order of magnitude larger than for the other schemes. They therefore compute at least an order of magnitude faster. Because of the simplicity and speed of computing the coefficients of the modified Tustin's method, this appears to be the scheme to be recommended for this type of feedback system. Note, however, that this scheme is accurate only when τ is at least a few times larger than Δ (see equations 3.4 and 3.7), and breaks down completely for $\tau' < 1$ (with $-a_1$ even becoming negative for $\tau' < 0.5$).

4.2 Photopigment Bleaching: Dynamics on Different Timescales. For an example of a stiff set of differential equations, we look at the dynamics of photopigment bleaching in human cones (Mahroo & Lamb, 2004; Lamb & Pugh, 2004; van Hateren & Snippe, 2007). For present purposes, a suitable form of the equations is

$$\dot{R} = [I(1 - B - R) - R]/\tau_R \tag{4.5}$$

$$\dot{B} = R/\tau_R - \frac{0.2}{B + 0.2} B/\tau_B. \tag{4.6}$$

Here I is a (scaled) light intensity, R is the (normalized) amount of photopigment excited by light, and B is the (normalized) amount of bleached photopigment. The rate by which excited pigment is bleached is governed by first-order kinetics ($1/\tau_R$), whereas the reconversion of bleached pigment to excitable pigment is governed by rate-limited dynamics (Mahroo & Lamb, 2004): the second term on the right-hand side of equation 4.6 is consistent with first-order kinetics for small B but saturates for large B. Equations 4.5 and 4.6 form a stiff set of equations because the time constants $\tau_R = 3.4 \cdot 10^{-3}$ s and $\tau_B = 25$ s differ substantially. Through the factor $(1 - B - R)$, bleaching provides a slow gain control, controlling the sensitivity of the eye in bright light conditions.

Rewriting the equations into the form of equation 2.2 gives

$$\tau_R \dot{R} = I\,(1 - B - R) - R \tag{4.7}$$

$$\tau_b \dot{B} = g_B R - B \quad \text{with} \quad \tau_b = \tau_B \frac{B + 0.2}{0.2} \quad \text{and} \quad g_B = \tau_b/\tau_R. \tag{4.8}$$

This processing scheme is depicted in Figure 5A, where τ_b and g_B at time t_n are derived from B at time t_{n-1}. Note that the phase advance of τ^- is sufficient for the loop involving τ_b, but provides only half of the required phase advance for the direct loop. Figure 5B shows a benchmark calculation using ode45 and the result of using the scheme of Figure 5A with the modified Tustin's method. The stimulus I steps at $t = 0$ from 10^{-5} to a sinusoidal modulation around 10^{-3}. Because an instantaneous step contains considerable power in its high-frequency components, using a recursive filter with a rather large Δ causes significant aliasing, which in this example would noticeably affect the response right after the step. To reduce the effect of aliasing, the step was assumed here to take 1 ms, that is, there is a linear taper between $t = 0$ and 1 ms. Figure 5C compares the rms error of the various schemes as a function of Δ. Again, the ZOH and the modified Tustin's method perform best, despite the fact that there is no complete compensation of the feedback delay.

4.3 Spiking Neurons: Hodgkin-Huxley Equations. As a final example of a highly nonlinear system with fast dynamics, we look at the Hodgkin-Huxley equations for spike generation (Hodgkin & Huxley, 1952). Following the formulation by Gerstner and Kistler (2002, Chapter 2.2), these equations are given by equations 4.9 to 4.12:

$$C\dot{u} = -g_{\mathrm{Na}} m^3 h (u - E_{\mathrm{Na}}) - g_{\mathrm{K}} n^4 (u - E_{\mathrm{K}}) - g_{\mathrm{L}} (u - E_{\mathrm{L}}) + I, \tag{4.9}$$

where u is the membrane potential (in mV, defined relative to the resting potential), C the membrane capacitance (taken as 1 μF/cm^2), the input

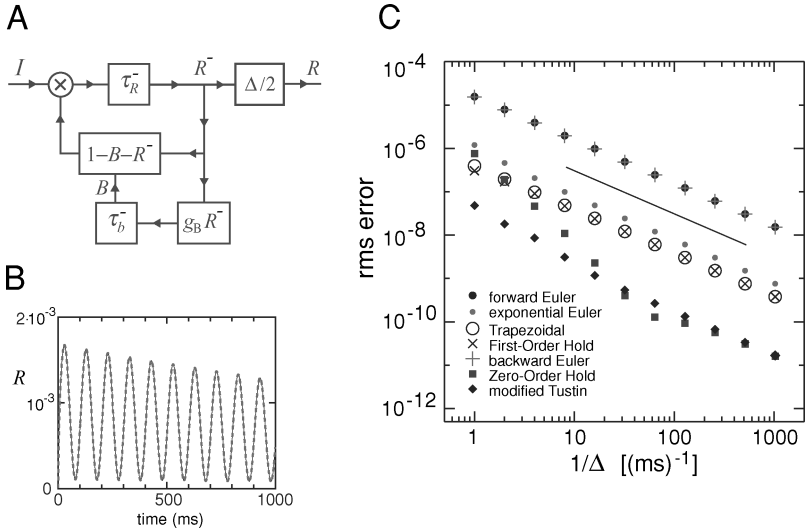

Figure 5: (A) System diagram of equations 4.7 and 4.8. (B) Thin continuous line: response R of equations 4.7 and 4.8, using $\tau_R = 3.4\,\text{ms}$ and $\tau_B = 25\,\text{s}$, to $I = 10^{-3}(1 + 0.9\sin(2\pi f t))$ for $t \geq 1\,\text{ms}$, $I = 10^{-5}$ for $t < 0$, and $I = 10^{-5} + (10^{-3} - 10^{-5})t$ for $0 \leq t < 1\,\text{ms}$, with $f = 10\,\text{Hz}$, computed with Matlab ode45; dashed line: result of filtering with the scheme of A, with $\Delta = 1\,\text{ms}$ and using the modified Tustin's method for τ^-. (C) Root-mean-square (rms) error between the various recursive filters used for the scheme of A and the result of ode45 at its maximum accuracy setting. Input as in B. The thin straight line has a slope of -1 in double-logarithmic coordinates.

variable I is externally applied current, and the other terms represent membrane currents (consisting of a sodium, potassium, and leakage current). The membrane currents are given by the reversal potentials for the ions (in mV, defined relative to the resting potential: $E_{\text{Na}} = 115$, $E_{\text{K}} = -12$, and $E_{\text{L}} = 10.6$), by conductances (in mS/cm^2, $g_{\text{Na}} = 120$, $g_{\text{K}} = 36$, and $g_{\text{L}} = 0.3$), and by variables n, m, and h, describing the gating of the ion channels by the membrane potential:

$$\dot{n} = \alpha_n(1 - n) - \beta_n n \tag{4.10}$$

$$\dot{m} = \alpha_m(1 - m) - \beta_m m \tag{4.11}$$

$$\dot{h} = \alpha_h(1 - h) - \beta_h h. \tag{4.12}$$

The rate constants α and β are functions of u, the form of which was determined empirically by Hodgkin and Huxley (1952): $\alpha_n = (0.1 - 0.01u)/[\exp(1 - 0.1u) - 1]$, $\beta_n = 0.125\exp(-u/80]$, $\alpha_m = (2.5 - 0.1u)/$

$[\exp(2.5 - 0.1u) - 1]$, $\beta_m = 4\exp(-u/18]$, $\alpha_h = 0.07\exp(-u/20)$, and $\beta_h = 1/[\exp(3 - 0.1u) + 1]$.

Rewriting the equations into the form of equation 2.2 gives

$$\tau_e \dot{u} = R_e(I + I_e) - u$$

$$\text{with} \quad I_e = g_{\mathrm{Na}} m^3 h E_{\mathrm{Na}} + g_{\mathrm{K}} n^4 E_{\mathrm{K}} + g_{\mathrm{L}} E_{\mathrm{L}}$$

$$R_e = 1/\left(g_{\mathrm{Na}} m^3 h + g_{\mathrm{K}} n^4 + g_{\mathrm{L}}\right) \quad \text{and} \quad \tau_e = R_e C \tag{4.13}$$

$$\tau_n \dot{n} = n_\infty - n$$

$$\text{with} \quad \tau_n = 1/(\alpha_n + \beta_n) \quad \text{and} \quad n_\infty = \alpha_n/(\alpha_n + \beta_n) \tag{4.14}$$

$$\tau_m \dot{m} = m_\infty - m$$

$$\text{with} \quad \tau_m = 1/(\alpha_m + \beta_m) \quad \text{and} \quad m_\infty = \alpha_m/(\alpha_m + \beta_m) \tag{4.15}$$

$$\tau_h \dot{h} = h_\infty - h$$

$$\text{with} \quad \tau_h = 1/(\alpha_h + \beta_h) \quad \text{and} \quad h_\infty = \alpha_h/(\alpha_h + \beta_h) \tag{4.16}$$

This processing scheme is depicted in Figure 6A. The feedback is partly additive (through the gated current I_e, which acts as a strong positive feedback during the rising phase of the spike and as a negative feedback during the potassium-driven afterhyperpolarization), partly multiplicative (through the input resistance R_e, which drops considerably during the spike and is the main cause of the absolute refractory period of the neuron), and partly through the time constant τ_e, causing fast dynamics during the spike. Note that the system contains, for each of the three feedback variables, two low-pass filters in series (τ_e and the one belonging to either n, m, or h); thus, we can fully use the phase advance of τ^- as in the example on phototransduction. Figures 6C and 6D show a benchmark calculation using ode45 of the response (continuous line) to a current input as shown in Figure 6B. This stimulus is again tapered at the beginning to reduce aliasing. Some tapering is realistic, because normally the axon of a spiking neuron (where spiking starts) will not be driven by instantaneous current steps, but only by band-limited currents because of low-pass filtering by the cell body and dendrites. Figure 6C shows the result of using the scheme of Figure 6A with trapezoidal (obviously without the $\Delta/2$ processing block), and Figure 6D with the modified Tustin's method. Figure 6E compares the rms error of the various schemes as a function of Δ. Again, the ZOH and the modified Tustin's method perform best. In particular, the modified Tustin's method provides accurate results: even at a course $\Delta = 1/2$ ms, it misses no spikes in the example of Figure 6, and the timing precision of the spikes is on the order of 0.1Δ. This contrasts with, for instance, a scheme like trapezoidal, which needs Δ at least as small as 1/32 ms in order not to miss spikes, and has a timing precision of the spikes on the order of 10Δ.

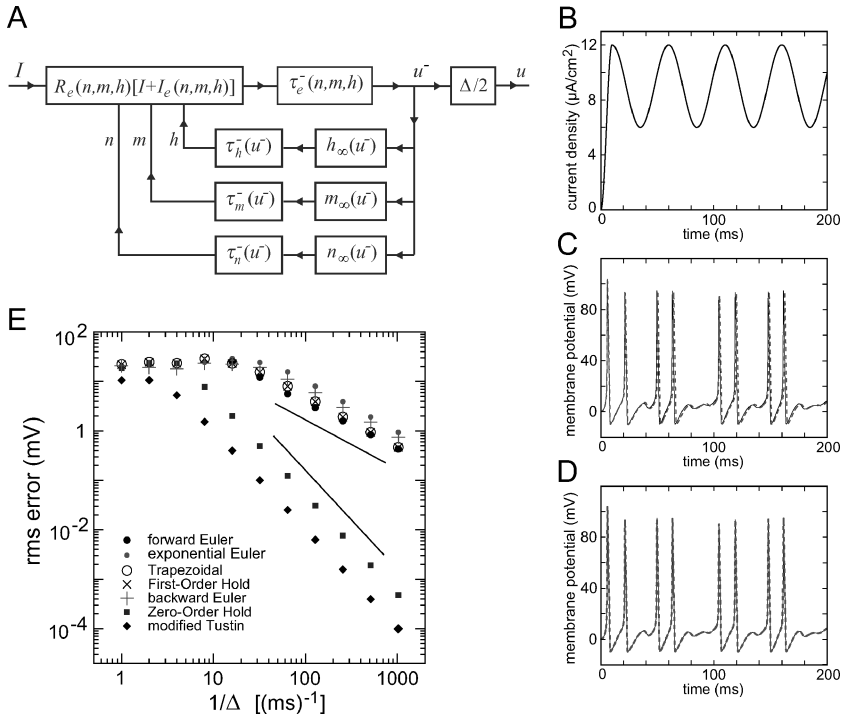

Figure 6: (A) System diagram of equations 4.13 to 4.16. (B) Driving current density I, with $I = 0$ for $t < 0$, $I = I_0 \sin^2(0.5\pi\, t/t_0)$ for $0 \le t < t_0$ ms, and $I = I_0(1 - 0.5\sin^2(0.5\pi f(t - t_0)))$ for $t \ge t_0$, with $t_0 = 10$ ms a taper, $f = 10$ Hz, and $I_0 = 12\,\mu$A/cm^2. (C) Thin continuous line: response u of equation 4.13 to the stimulus defined at B, computed with Matlab ode45; dashed line: result of filtering with the scheme of A, with $\Delta = 1/32$ ms and using trapezoidal for τ. (D) Thin continuous line: as in C; dashed line: result of filtering with the scheme of A, with $\Delta = 1/32$ ms and using the modified Tustin's method for τ^-. (E) Root-mean-square (rms) error between the various recursive filters used for the scheme of A and the result of ode45 at its maximum accuracy setting. Input as in B. The thin straight lines have slopes of -1 and -2 in double-logarithmic coordinates.

4.4 When to Use τ^- or τ^0. Two of the examples given above involve feedback with exactly two low-pass filters in the forward and backward branches of the feedback loop. For these schemes low-pass filters with phase advance are clearly useful. However, for other topologies, this is not necessarily the case. Figure 7 shows a few examples. When concatenating low-pass filters and static nonlinearities (see Figure 7A), zero-delay filters τ^0 may be used as an alternative to using τ^- and performing delay correction at a later stage. In a feedforward structure as shown in Figure 7B, a zero-delay

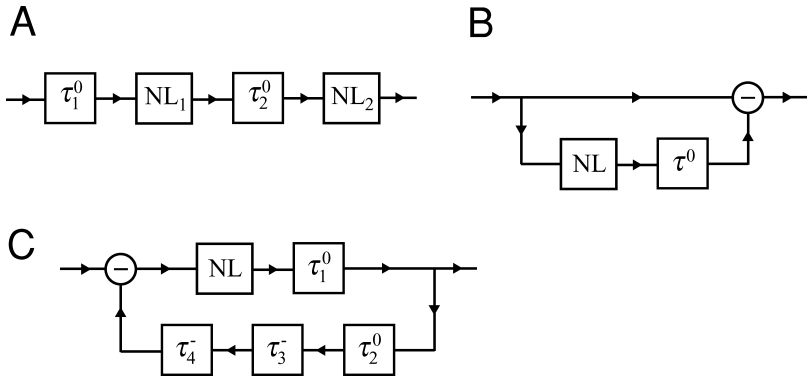

Figure 7: (A) Concatenation of low-pass filters and nonlinearities (NL), where zero-delay low-pass filters can be used. (B) In a feedforward loop as shown, a zero-delay low-pass filter should be used. (C) In a feedback loop, the total delay compensation needs to match the implicit delay Δ of the computational feedback scheme.

filter must be used. Similarly, if a feedback scheme contains more than two low-pass filters, some of the filters need to be zero delay (see Figure 7C).

If a system contains a feedback loop with only one low-pass filter in either the feedforward or feedback branch, a filter τ^- can provide only half of the required phase advance. In those situations, as in the example on photopigment bleaching given above, it is still helpful to use τ^-, in addition to making Δ sufficiently small. In principle, a phase advance (a delay of $-\Delta/2$) might be added by implementing it as a linear extrapolation $y_n = 1.5x_n - 0.5x_{n-1}$. However, I have not tested such a scheme, which might have stability problems.

Finally, if a feedback loop contains no low-pass filters at all, it is in fact identical to a static nonlinearity and can usually be treated analytically or by a precomputed lookup table.

4.5 Comparison with a Fourth-Order Runge-Kutta Integration Scheme. Although this letter focuses on simple autoregressive filters working on data with a given step size, it is interesting to compare the performance of the scheme with a standard integration method, such as fourth-order Runge-Kutta (RK4; Press et al., 1992). Figure 8 shows the results for RK4 and the modified Tustin's method, applied to a fairly complex model of the macaque retinal horizontal cell (van Hateren, 2005). This model consists of cones connected to horizontal cells in a feedback circuit and constitutes a cascade of a static nonlinearity, two nonlinear (divisive) feedback loops, and a subtractive feedback loop. All loops contain, in various configurations, low-pass filters and static nonlinearities. For details, such as parameter values and the differential equations involved, see van Hateren (2005).

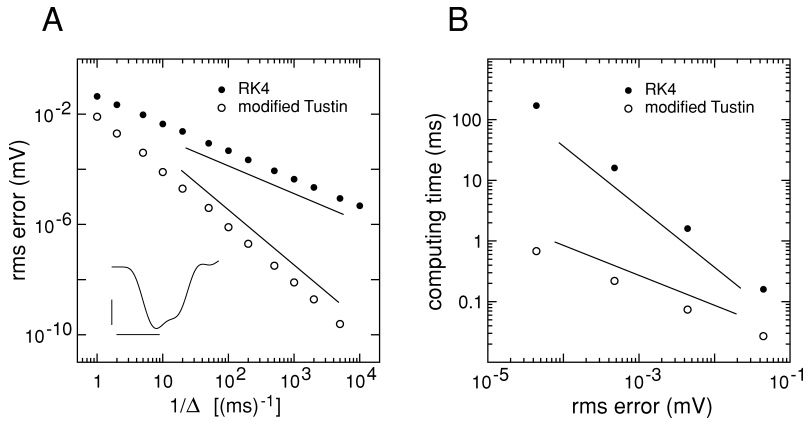

Figure 8: (A) Root-mean-square (rms) error of computing the response (inset, vertical bar = 2 mV) to a 40 ms light flash (horizontal bar inset) of the macaque retinal horizontal cell model of van Hateren (2005). Both a fourth-order Runge-Kutta scheme (RK4, fixed time step, routines rkdumb/rk4 of *Numerical Recipes*, Press et al., 1992; the input is an analytical block function according to the horizontal bar) and modified Tustin were implemented in a double-precision Fortran90 program (Intel compiler, Linux, 3.0 GHz Xeon). Errors are calculated relative to the result of modified Tustin at a time step $\Delta = 0.1\ \mu$s. The straight lines have slopes of -1 and -2 on double-logarithmic coordinates. (B) Computing times for RK4 and modified Tustin at matched rms error. For the four sets of data points, the time steps Δ for (RK4, modified Tustin) are (1 μs, 70 μs), (10 μs, 230 μs), (0.1 ms, 0.7 ms), and (1 ms, 2.5 ms). Ratios of computing times are 250, 70, 20, and 6. The straight lines have slopes of -1 and -0.5 on double-logarithmic coordinates.

The inset in Figure 8A shows the response of the model horizontal cell to a 40 ms light flash (horizontal bar) of contrast 2 given on a background of 100 td (see van Hateren, 2005, for details on the stimulus). The vertical scale bar denotes 2 mV. This model was computed using either modified Tustin for the components (as in the examples in this article) or RK4 for the entire set of differential equations. It should be stressed that this use of RK4 is different from the use of integrators, such as forward Euler, earlier in this letter, where each low-pass filter was integrated separately. Here the RK4 algorithm is used in the conventional way on the entire model at once. All rms errors are calculated relative to the result of modified Tustin at a step size of 0.1 μs. Identical results were obtained when calculating all errors relative to RK4 at 0.1 μs, be it that errors then saturate at (i.e., do not go below) $4.7 \cdot 10^{-6}$ because of the limited accuracy of RK4 at 0.1 μs. Figure 8A shows the rms error of RK4 and modified Tustin. For all step sizes shown, modified Tustin

outperforms RK4. The different scaling behavior is indicated by the two lines with slopes of -1 and -2 on the double-logarithmic coordinates.

As argued by Morrison, Straube, Plesser, and Diesmann (2007), in many situations, the most interesting measure of performance of an integration method is the computing time required to achieve a given accuracy. This is shown in Figure 8B for the two methods considered here. For this calculation, the step size of modified Tustin was adjusted such that the accuracy of the result matched one of the RK4 calculations, and the corresponding computing times of the methods are plotted. Depending on accuracy, modified Tustin is typically one to two orders of magnitude faster than RK4. It should be noted that the calculation at the largest rms error already required a step size for modified Tustin (2.5 ms) that brought it well out of the range where the condition that the step size should be a few times smaller than τ (see equations 3.4 and 3.7) is valid, because the fastest low-pass filters in the model have time constants of 3 to 4 ms (van Hateren, 2005). Nevertheless, even under these conditions, modified Tustin is approximately six times faster than RK4 at the same accuracy.

5 Discussion

The fast recursive scheme presented in this letter is particularly suited for situations where computing time is restrictive, for example, when large arrays of neurons need to be computed. The scheme is fast because each component is updated at each time step with only a few floating-point operations. The examples given show that it is already quite accurate with fairly large time steps. It accomplishes this by computing feedback in a way that makes use of the fact that several autoregressive implementations of first-order low-pass filters produce an implicit phase advance of half a sample distance. The computational scheme is associated with a simple diagrammatic representation that makes it relatively easy to get an intuitive understanding of the dynamics and the processing flow and allows convenient symbolic manipulation (e.g., rearranging modules into equivalent schemes).

Because the τ of the low-pass filters may depend on input and system variables, the filter coefficients may require updating at each time step. This may constitute a significant part of the computational load. Fortunately, the coefficients for the trapezoidal rule (for τ^0) and the modified Tustin's method (for τ^-) can be obtained with only a few floating-point operations. These schemes also give results at least as accurate as any of the other schemes and therefore should be considered as the first choice.

The scheme presented here is primarily intended for nonlinear filtering. It could be used for arbitrary linear filtering as well, because any linear filter can be approximated by a parallel arrangement of a number of low-pass filters with different weights and time constants. However, I have not tested how well it performs on such arrangements, and it seems likely that there are

better ways to deal with arbitrary linear filters. One possibility is to use the matrix exponential (Rotter & Diesmann, 1999), which is particularly suited when the signal consists of (or can be approximated by) point processes, as is common in calculating networks of spiking neurons. The matrix exponential can also be viewed as equivalent to a ZOH model and then needs a $\Delta/2$ compensation depending on whether it is used in a feedforward branch or as part of a nonlinear feedback branch. Another possibility is to use canned routines, like c2d in Matlab, that provide coefficients for a recursive discrete system corresponding to any rational continuous transfer function. For a linear filter that is part of a nonlinear feedforward loop, the c2d routines using FOH or Tustin's method are required, whereas ZOH is required when the linear filter is part of a feedback loop and a phase advance is wanted.

All calculations presented in this letter were done with double-precision arithmetic. For strongly stiff problems, such a precision is necessary because of the large difference in time constants; the time step needs to be small enough to accommodate the shortest time constant, but such a short time step results in considerable error buildup in the processing of the largest time constant if single-precision arithmetic is used. However, I found that for the examples discussed in this letter, single precision arithmetic already gives accurate results. This is of interest because using single precision may accelerate computation, depending on processor architecture. Moreover, stream processors such as GPUs may not yet support double-precision arithmetic (although double precision can be readily emulated Göddeke et al., 2007—and GPUs with double precision are announced for early 2008).

I found that simulating the response of a large array of cones using the cone model of van Hateren and Snippe (2007), of which the examples of sections 4.1 and 4.2 are part, provides performance one to two orders of magnitude higher on current GPUs than on current CPUs. Such performance is of interest for developing and testing models of the human retina (van Hateren, 2007) and also for using light adaptation in human cones as an algorithm for rendering and compression high-dynamic range video (van Hateren, 2006).

Acknowledgments

I thank Sietse van Netten and Herman Snippe for comments on the manuscript.

References

Ahrenberg, L., Benzie, P., Magnor, M., & Watson, J. (2006). Computer generated holography using parallel commodity graphics hardware. *Optics Express*, *14*, 7636–7641.

Bechhoefer, J. (2005). Feedback for physicists: A tutorial essay on control. *Rev. Mod. Phys.*, *77*, 783–838.

Bower, J. M., & Beeman, D. (1998). *The book of GENESIS: Exploring realistic neural models with the GEneral NEural SImulation System* (2nd ed.). New York: Springer-Verlag.

Brown, K. S. (2000). *Lead-lag algorithms*. Available online at http://www.mathpages.com/home/kmath198/kmath198.htm.

Butera, R. J., & McCarthy, M. L. (2004). Analysis of real-time numerical integration methods applied to dynamic clamp experiments. *J. Neural Eng., 1*, 187–194.

Gerstner, W., & Kistler, W. M. (2002). *Spiking neuron models*. Cambridge: Cambridge University Press.

Göddeke, D., Robert Strzodka, R., & Turek, S. (2007). Performance and accuracy of hardware-oriented native-, emulated- and mixed-precision solvers in FEM simulations. *International Journal of Parallel, Emergent and Distributed Systems*, *22*, 221–256.

Guerrero-Rivera, R., Morrison, A., Diesmann, M., & Pearce, T. C. (2006). Programmable logic construction kits for hyper-real-time neuronal modeling. *Neural Comp., 18*, 2651–2679.

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol., 117*, 500–544.

Lamb, T. D., & Pugh, E. N. (2004). Dark adaptation and the retinoid cycle of vision. *Progr. Ret. Eye Res., 23*, 307–380.

MacGregor, R. J. (1987). *Neural and brain modeling*. San Diego: Academic Press.

Mahroo, O. A. R., & Lamb, T. D. (2004). Recovery of the human photopic electroretinogram after bleaching exposures: Estimation of pigment regeneration kinetics. *J. Physiol*., *554*, 417–437.

Mead, C. (1989). Analog *VLSI and neural systems*. Reading, MA: Addison-Wesley.

Moore, J. W., & Ramon, F. (1974). On numerical integration of the Hodgkin and Huxley equations for a membrane action potential. *J. Theor. Biol*., *45*, 249–273.

Morrison, A., Straube, S., Plesser, H. E., & Diesmann, M. (2007). Exact subthreshold integration with continuous spike times in discrete-time neural network simulations. *Neural Comp*., *19*, 47–79.

Oppenheim, A. V., & Schafer, R. W. (1975). *Digital signal processing*. Englewood Cliffs, NJ: Prentice Hall.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in Fortran*. Cambridge: Cambridge University Press.

Rotter, S., & Diesmann, M. (1999). Exact digital simulation of time-invariant linear systems with applications to neuronal modeling. *Biol. Cybern*., *81*, 381–402.

Rush, S., & Larsen, H. (1978). A practical algorithm for solving dynamic membrane equations. *IEEE Trans. Biomed. Eng., 36*, 389–392.

Thong, T., & McNames J. (2002). Transforms for continuous time system modeling. In *Proceedings of the 45th IEEE Midwest Symposium on Circuits and Systems* (pp. II-408–II-411). Piscataway, NJ: IEEE Press.

Unser, M. (1999). Splines—A perfect fit for signal and image processing. *IEEE Signal Process*. *Mag*., *16*, 22–38.

Unser, M. (2005). Cardinal exponential splines: Part II—Think analog, act digital. *IEEE Trans. Signal Process., 53*, 1439–1449.

van Hateren, J. H. (2005). A cellular and molecular model of response kinetics and adaptation in primate cones and horizontal cells. *J. Vision*, *5*, 331–347.
van Hateren, J. H. (2006). Encoding of high dynamic range video with a model of human cones. *ACM Transactions on Graphics*, *25*, 1380–1399.
van Hateren, J. H. (2007). A model of spatiotemporal signal processing by primate cones and horizontal cells. *J. Vision*, *7*(3), 3, 1–19.
van Hateren, J. H., & Snippe, H. P. (2007). Simulating human cones from mid-mesopic up to high-photopic luminances. *J. Vision*, *7*(4), 1, 1–11.

Received April 12, 2007; accepted August 29, 2007.